HOMELAB

SZOFTVER

A BASIC JELKÉSZLETE:

- számok: 0 1 2 3 4 5 6 7 8 9
- 31 db magyar kis- és nagybetű
- műveleti jelek: + - / *
- relációk: < > =
- egyéb jelek: ! " # $ % & ' ( ) , . : ; ?

A BASIC által használt speciális karakterek a következők:

CR — CARRIAGE RETURN. A begépelt sort zárja le.

CLS — Clear. Shift/CR-rel érhető el; a képernyőt törli.

HOME — HOME. Shift/space-vel érthető el; a cursort a kép tetejére teszi.

F1/F2 — Az F1 és F2 funciójú gombok egyidejű lenyomásával érhető el a BREAK, amely a BASIC program futását szakitja meg. Más gépeken általában a CONTROL/C. Csak a program futása közben érvényes.

CURSOR — Mozgató karakterek. A CURSOR segitségével a képernyő bármely pontja elérhető, ott javitás végezhető vagy a szöveg újrairható. A CR hatására mindig az a sor kerül bevételre, amelyikben a CURSOR éppen volt. Itt logikai sor értendő, nem fizikai!

INS — INSERT. A shift/→ segitségével érhető el, új karakter beszúrására alkalmas. A CURSOR-tól jobbra eső szövegrészt eggyel jobbra lépteti, igy a CURSOR helyén egy szabad karakter marad. Ismételt megnyomással újabb betükhöz karakterek keletkeznek, amelyek helyébe új szöveg irható be.

DEL — DELETE. A shift/← segitségével érhető el, a egy karakter törlésére szolgál. A CURSOR-tól jobbra eső szövegrészt eggyel balra lépteti úgy, hogy a CURSOR által kijelölt karakter elvész. Ha a CURSOR sor végén van, az utolsó karaktert törli.

A BASIC JELKÉSZLETE:

- számok: 0 1 2 3 4 5 6 7 8 9
- 31 db magyar kis- és nagybetű
- műveleti jelek: + - / *
- relációk: < > =
- egyéb jelek: ! " # $ % & ' ( ) , . : ; ?

A BASIC által használt speciális karakterek a következők:

CR — CARRIAGE RETURN. A begépelt sort zárja le.

CLS — Clear. Shift/CR-rel érhető el; a képernyőt törli.

HOME — HOME. Shift/space-vel érthető el; a cursort a kép tetejére teszi.

F1/F2 — Az F1 és F2 funciójú gombok egyidejű lenyomásával érhető el a BREAK, amely a BASIC program futását szakitja meg. Más gépeken általában a CONTROL/C. Csak a program futása közben érvényes.

CURSOR — Mozgató karakterek. A CURSOR segitségével a képernyő bármely pontja elérhető, ott javitás végezhető vagy a szöveg újrairható. A CR hatására mindig az a sor kerül bevételre, amelyikben a CURSOR éppen volt. Itt logikai sor értendő, nem fizikai!

INS — INSERT. A shift/→ segitségével érhető el, új karakter beszúrására alkalmas. A CURSOR-tól jobbra eső szövegrészt eggyel jobbra lépteti, igy a CURSOR helyén egy szabad karakter marad. Ismételt megnyomással újabb betükhöz karakterek keletkeznek, amelyek helyébe új szöveg irható be.

DEL — DELETE. A shift/← segitségével érhető el, a egy karakter törlésére szolgál. A CURSOR-tól jobbra eső szövegrészt eggyel balra lépteti úgy, hogy a CURSOR által kijelölt karakter elvész. Ha a CURSOR sor végén van, az utolsó karaktert törli.

BELL — BELL. Shift/↓ érhető el. Hatására a gép rövid hangjelzést ad. Stringbe beépitve igy fütty is printelhető.

TAB — Tabulátor. Shift/↑ érhető el. Hatására a cursor új zóna elejére lép, és az útjában levő karaktereket törli.

ALT — ALTER. Ez a gomb a nagybetüs és kisbetüs irásmód felcserélésére szolgál, csak a betükre van hatással.
Ha a cursor teli négyzet alakjában villog, akkor a leütött gombok nagybetüt adnak, és a shift vált kisbetüre. Az ALT ezt úgy változtatja meg, hogy egy üres négyzet fog villogni, és alapesetben kisbetü, shift-tel pedig nagybetü lesz. Ujabb ALT visszavált az előbbi üzemmódra és igy tovább.

SPACE — Soremelésnél, ha teli a kép és ezt a gombot nyomva tartjuk, egyenként lassan lépteti a sorokat. Ha shift-tel együtt nyomjuk meg, teljesen leáll a kiirás, egészen addig, mig újra shift/space-t nem nyomunk. Amig áll a kép, addig is van lehetőség a sorokat egyenként léptetni a SPACE nyomogatásával. A shift/CR ebben az állapotban egy teljes új képet hoz be. Mindezeknek listázásnál és folyamatosan keletkező eredményeknél van szerepe.

: — Egy BASIC sorban tetszőleges számú utasitás lehet, ezek között kettőspont az elválasztó karakter.

, — Lista elemeket választ el vagy a PRINT utasitás formátumát határozza meg.

— PRINT, INPUT és LIST utasitásoknál OUTPUT, illetve INPUT eszközt jelöl ki.

' — Feltételes listák elválasztó karaktere /lásd ON, IF/, vagy a BASIC sor hátralevő részének letiltására használható. Az ' mögött álló utasitások -az IF és az ON kivételével- sohasem hajtódnak végre. IF vagy ON után nem szerepelhet szövegkonstansban.

? Ekvivalens a PRINT kulcsszóval.

" Stringkonstansok elejét és végét jelzi. Az " között lévő , . ; : ? karakterek elvesztik előbb felsorolt jelentésüket, ilyenkor ezek is hozzátartoznak a szövegkonstanshoz.

A SZÁMÁBRÁZOLÁSRÓL

A gép belső számábrázolása 4 bájt-os lebegőpontos. Ennek megfelelően +1E38 és +1E-38 közé eső számokat tud ábrázolni, 1E-7 relativ hibakorláttal. A gép a számokat 6 jegyre kerekítve írja ki 999999 és 0.1 tartományban törtalakban, azon kívül pedig normál alakban. A mantissza és a karakterisztika + előjele, valamint a tizedespont előtti érvénytelen nullák elhagyhatók.
A duplapontos bővítésben a számok belső ábrázolása 16 jegyre nő és a gép 14 jegyet ír ki.

VÁLTOZÓK

A BASIC-ben használt változók azonosítója -típustól függetlenül- legfeljebb két betű lehet. A 30 nagybetűn kívül más karakter nem használható. Több betűből álló azonosítók esetében csak az első két betűt jegyzi meg és veszi figyelembe. A változók neve nem tartalmazhatja a kulcsszavak karaktersorozatát. A változók tartalmuk szerint REAL vagy STRING típusúak lehetnek. Ez utóbbinál az azonosító után áll.
Szervezésük szerint a változók lehetnek egyszerű változók, vektorok és mátrixok. A vektorok és mátrixok /tömbök/ azonosítója után zárójelben áll az egy-, ill. kéttagú kifejezés. Ugyanolyan típusú vektorok és mátrixok azonosítására nem használhatók azonos betűkombinációk. A legnagyobb tömbindex 255 lehet.

A BASIC-ben van öt speciális változó is, nevezetesen:

PI, INKEY, INKEY$, CR, HM.

PI A π értékét adja: PI=3.14159

INKEY, INKEY$ Ez a változó lehet REAL és STRING típusú is. Mindkettő az éppen lenyomott billentyűt adja: INKEY a lenyomott gomb ASCII kódját, az INKEY$ pedig

magát a karaktert. Ha nincs lenyomva semmi, a5.
INKEY=Ø és INKEY$ üres string.
Az F1, F2 és SHIFT gombok lenyomását a gép külön nem veszi észre.

CR COLOR. A képernyőn PLOT utasítással megjeleníteni kívánt pont színét határozza meg. Ha CR=Ø fekete, ha CR=1 fehér a pont. A CR=2 invertálást jelent. CR-t csak akkor kell állítani, ha az éppen benne lévő szín nem megfelelő.

HM HIGHEST MEMORY. Az ide beírt decimális szám a BISIC számára felhasználható legmagasabb memóriahely címét jelenti. Ezzel a változóval gépikódú programrészletek és adatmezők számára rezervált terület.

KIFEJEZÉSEK

A kifejezésekre ugyanaz vonatkozik, mint a standard BASIC-ben. A prioritási elv és a balról-jobbra szabály érvényesül. Kifejezésekben használhatók az AND, OR, NOT műveletek és az összes relációk is. Az AND, OR, NOT logikai műveletek a számok egészrészének 2 bájtos fixpontos kettes komplemens alakjának bitjeire vonatkoznak. Pl.
63 AND 16=16; 4 OR 2=6; NOT 1=-2
A relációknak szintén értékük van. Ha egy reláció hamis, akkor az értéke Ø, ha igaz értéke 1./A=2 esetén: (A=2)=1; (A=57)=Ø; (A>Ø)=1/

STANDARD FÜGGVÉNYEK

Kevés kivételtől eltekintve a HOMELAB-BASIC az összes standard függvényt tartalmazza.

A függvények a következők:

ABS(X)	INT(X)	RND(X)	SGN(X)	SIN(X)
COS(X)	TAN(X)	ATN(X)	SQR(X)	EXP(X)
LOG(X)	FRE(X)	USR(X)	PEEK(X)	POINT(X,Y)

ABS(X) X kifejezés abszolút értékét adja.
ABS(1)=1; ABS(-2)= 2

INT(X) X kifejezés egész részét adja.
INT(4.5)=4; INT(4)=4; INT(-3.2)=-4

SGN(X) X kifejezés előjelét adja. Értéke +1, Ø,-1 lehet.
SGN(-3)=-1; SGN(Ø)=Ø; SGN(4.1)=1.

SIN(X) X kifejezés sinusát adja; X értéke radiánban értendő.

COS(X) X kifejezés cosinusát adja; X értéke radiánban értendő.

TAN(X) X kifejezés tangensét adja; X értéke radiánban értendő.

ATN(X) X kifejezés arcus tangensét adja. A függvény értéke radiánban adja a szöget.

SQR(X) X kifejezés négyzetgyökét adja. X értéke nem lehet negatív szám.

EXP(X) e-t, a természetes logaritmus alapját; az X kifejezés által adott hatványra emeli.

LOG(X) X kifejezés természetes alapú logaritmusát képezi.

RND(X) Ha X<Ø, akkor ABS(X) a véletlenszám-generátor kezdőértéke lesz. Nagyságára nézve nincs kikötés. Mivel RANDOMIZE utasítás nincs ebben a BASIC-ben, ez a tulajdonság használható véletlen sorozatok reprodukálására.
Ha X>Ø, akkor Ø<RND(X)<X. X a véletlenszám intervalluma.
Ha X=Ø, akkor RND(X) az a szám, amiből a következő véletlenszám generálódni fog.

PEEK(X) Megadja az argumentumába írt kifejezés számértékének megfelelő című memóriahely tartalmát.
Ha X<8192 /2ØØØH/, akkor a PEEK nem memóriaterületet szólít meg, hanem az adott címmel IN gépikódú utasítást hajt végre.

Ily módon INPUT perifériák kezelhetők /több mint 8000 darab!/.
Ha X nagyobb mint 65536, akkor X értékéből ez a szám levonódik, és a PEEK a második lap megfelelő címét olvassa.

USR(X) Gépikódú szubrutinhivást tesz lehetővé adatátvitellel. A szubrutincimet előzőleg POKE utasitással kell beállitani az USR-vektorba. A gépikódú szubrutin magas cimbájtjának helye 16385, az alacsonyé pedig 16384. X egészrésze 2 bájtos fixpontos alakban a HL regiszterpárba kerül. RET gépikódú utasitás hatására a gépikódú rutin visszatér a BASIC-be, és USR(X) értéke a HL-ben lévő szám lesz.
A CPU összes regisztere felhasználható IY kivételével.

FRE(X) Megadja a szabad memóriaterület nagyságát. X értéke közömbös, de nem hiányozhat.

POINT(X,Y) Az X,Y koordinátájú képpont szinét adja meg. Értéke 0, ha a pont fekete, 1, ha fehér. A 0, 0 pont a képernyő bal alsó sarka, és 0≤X≤127 vagy 63; 0≤Y≤95.

STRINGEK, STRINGFÜGGVÉNYEK

Egy STRING hossza nem lehet nagyobb 255-nél és összesen 256-féle karakter fordulhat elő benne. Ezek mindegyike beirható a billentyüzetről is /a SHIFT, F1, F2 és egy jelgomb valamilyen kombinációjával/, és a CHR függvény segitségével is.
Van néhány karakter, amelynek speciális "nyomtatási képe" van. Ilyen, a karakterenként is kiadható: INS, DEL, BELL, TAB, HOME, CURSOR-mozgatások. Ezek a karakterek a billentyüzetről is beirhatók és beépithetők a stringkonstansokba, mert az F1 vagy F2 hatására elvesztik speciális tulajdonságaikat, és csak a jelük látszik. Kinyomtatáskor természetesen mint speciális karakterek iródnak ki.

A HOMELAB BASIC-ben az alábbi stringfüggvények találhatók:

ASC(X$)
Megadja az X$első karakterének kódját. Ha ez a karakter az ASCII készlet eleme, akkor a kód egyben ASCII kód is.

CHR$(I1, I2,...IN)
Azt a stringet adja meg, amely karaktereinek kódjai rendre: I k. I lehet speciális karakterek kódja is. ASCCHR(I) = I.

LFT$(X$, I)

X$első I karakteréből álló stringet adja meg.

RGH$(X$, I)

X$utolsó I karakteréből álló stringet adja meg.

MID$(X$, I, J)

X$I-edik karakterénél kezdődő és J darab karakterből álló stringet adja meg.

LEN(X$)

X$ hosszának számértékét adja meg.

STR$(X)

Azt a stringet adja meg, amelyik az X kifejezés értékének nyomtatási képe. Pl. X=3.1 STR$(X)="␣3.1"

VAL(X$)

Ha X egy számot ábrázol, vagy aritmetikai kifejezés stringképe, akkor VAL(X$)ennek a számértékét adja.
Pl. A=3 és B=4 esetén VAL("A*4+64*B")= 268, vagy
VAL("238.45") = 238.45 , VAL ("") = ø
A stringekre értelmezve vannak a relációk és az összeadás is. Az összeadás egyszerű egymás után írást jelent. A$=B$, ha hosszuk egyenlő és elemeik rendre megegyeznek. A$ < B$, ha B$-ben előbb van nagyobb kódú karakter, mint A$-ben, illetve A$ kezdőstringje B$-nek. /B$ az elején tartalmazza A$-et/.

Az alább felsorolt kulcsszavak a CONT kivételével utasításként és parancsként egyaránt kiadhatók. /CONT -nem lehet utasitás./ Az utasitások zöme teljesen azonos a szabványos BASIC-kel.

Az utasitások és parancsok kulcsszókészlete:

BEEP	CALL	CONT	CUR	DATA	DIM
END	EXT	FOR	GOSUB	GOTO	IF
INPUT	LIST	LOAD	MON	NEXT	NEW
ON	PLOT	POKE	POP	PRINT	READ
REM	RESTORE	RETURN	RUN	SAVE	STEP
THEN	TO				

BEEP A$

Hankeltő utasitás, elfütyüli a mögötte álló stringkifejezést. A strinbe az egyes karakterek különböző ritmusokat és hangmagasságoknak felelnek meg.
A karakterek jelentése a következő:

0 -	31	érvénytelen
32		zárókarakter, ez feltétlenül kell a string végén.
33 -	63	ritmust határoz meg. 33 a leggyorsabb sebesség, 63 a leglassubb sebesség.
64 -	255	hangmagasság. A 64 a legmagasabb, 255 a legmélyebb.

CALL X1, X2,...

Gépikódú szubrutin hivások az X1, X2 stb. /decimális/ címekre. A szubrutinhivások a címsorrendnek megfelelően egymás után következnek. A címek között vesző az elválasztó jel. Címnek nem csak szám, hanem aritmetikai kifejezés is irható. A szubrutin gépikódú RET utasitással tér vissza a BASIC-be. A CPU regiszterei IY kivételével felhasználhatók.

CONT

BRK-val megszakitott program folytatását irja elő. Ez a folytatás csak akkor lehetséges, ha a BREAK üzenet óta még nem történt programsorbeirás, vagy bármilyen hiba üzenet. Ellenkező esetben CN ERROR keletkezik.

FOR X = A TO B STEP C: ....: :NEXT

Ciklusképző utasitás. X-a ciklus-változó, ami lehet vektor, vagy mátrix-elem is. A, B, C tetszőleges aritmetikai kifejezések. A ciklus-változó kezdőértéke A. B a ciklus végét adja, C pedig a növekmény. STEP C hiányozhat, ekkor a növekmény 1.
Ezután az un. deklaratiy rész után következik a ciklusmag, majd a NEXT cikluszáró utasitás. A ciklusváltozó értéke mindig a NEXT utasitás alatt módosul a növekménynyel. Ha C > Ø a ciklus addig tart, mig X ⩾ B. Ha C < Ø a ciklus X < B-ig jut. Ha a ciklus lejárt, a NEXT-et követő utasitás hajtódik végre.
Ciklusok tetszőleges mélységig egymásba skatulyázhatók.
Figyelem! A NEXT utasitás után nem szabad odairni a ciklusváltozót . Ezt más Basic-ek megengedik, vagy megkövetelik, de itt ez hibás !
Viszont egy NEXT-tel több ciklus is lezárható. A NEXT utasitás mögé irt minden vessző újabb ciklust zár le. /pl. NEXT,, ekvivalens a NEXT : NEXT : NEXT-el./
FOR ciklus nélkul használt NEXT esetén Pp Error-t uzen.

GOTO X

Feltétlen vezérlésátadó utasitás. A program végrehajtása, az X sornál folytatódik. X lehet egy sorszám is, de lehet aritmetikai kifejezés is. Ezzel kiszámitott GOTO képezhető. Ha X aritmetikai kifejezés, akkor nem kezdődhet számmal, vagy ha ez elkerülhetetlen, akkor zárójelbe kell tenni.

GOSUB X

Szubrutinhivás az X cimre. X-re ugyanaz érvényes, mint a GOTO utasitásnál. Lehetőség van egymás után több szubrutin meghivására is, ekkor a cimeket vesszővel elválasztva kell felsorolni. A szubrutinok értelemszerüen a cimek sorrendjében hajtódnak végre.

IF

Lásd az ON után !

FOR X = A TO B STEP C: ....: :NEXT

Ciklusképző utasitás. X-a ciklus-változó, ami lehet vektor, vagy mátrix-elem is. A, B, C tetszőleges aritmetikai kifejezések. A ciklus-változó kezdőértéke A. B a ciklus végét adja, C pedig a növekmény. STEP C hiányozhat, ekkor a növekmény 1.
Ezután az un. deklaratiy rész után következik a ciklusmag, majd a NEXT cikluszáró utasitás. A ciklusváltozó értéke mindig a NEXT utasitás alatt módosul a növekménynyel. Ha C > Ø a ciklus addig tart, mig X ⩾ B. Ha C < Ø a ciklus X < B-ig jut. Ha a ciklus lejárt, a NEXT-et követő utasitás hajtódik végre.
Ciklusok tetszőleges mélységig egymásba skatulyázhatók.
Figyelem! A NEXT utasitás után nem szabad odairni a ciklusváltozót . Ezt más Basic-ek megengedik, vagy megkövetelik, de itt ez hibás !
Viszont egy NEXT-tel több ciklus is lezárható. A NEXT utasitás mögé irt minden vessző újabb ciklust zár le. /pl. NEXT,, ekvivalens a NEXT : NEXT : NEXT-el./
FOR ciklus nélkul használt NEXT esetén Pp Error-t uzen.

GOTO X

Feltétlen vezérlésátadó utasitás. A program végrehajtása, az X sornál folytatódik. X lehet egy sorszám is, de lehet aritmetikai kifejezés is. Ezzel kiszámitott GOTO képezhető. Ha X aritmetikai kifejezés, akkor nem kezdődhet számmal, vagy ha ez elkerülhetetlen, akkor zárójelbe kell tenni.

GOSUB X

Szubrutinhivás az X cimre. X-re ugyanaz érvényes, mint a GOTO utasitásnál. Lehetőség van egymás után több szubrutin meghivására is, ekkor a cimeket vesszővel elválasztva kell felsorolni. A szubrutinok értelemszerüen a cimek sorrendjében hajtódnak végre.

IF

Lásd az ON után !

INPUT # X;A,B .....

Ez az INPUT-tal azonos módon működik, mindössze az a különbség, hogy az input-periféria nem a billentyűzet lesz. X szám vagy kifejezés azonosítja az input eszközt. # Ø-a billentyűzet. /Ez, mint az előbb is láttuk, el is hagyható./ További input perifériák kiszolgálására való rutinokat külön kell megadni. Ezt a gépikódú rész az INPUT # VEKTOR kapcsán ismerteti.

LIST

Kilistázza a programot. Ha utána egy szám áll, akkor csak azt a sort írja ki. Ha két szám áll kötőjellel, akkor a két sorszám közti területet írja ki. Itt a kettő közül bármelyik szám hiányozhat, ekkor értelemszerűen az adott sortól ill. az adott sorig listáz. Ha a LIST után # áll, akkor az előzővel azonos módon, de a printerre listáz.

LOAD " NEV "

Kazettás magnóról programot olvas. A rekord nevét idéző jelek közé kell tenni. Ha a név hiányzik, akkor az első érvényes rekordot veszi be. Ha értelmes rekordot talál, kiírja a nevet. Ha a név nem egyezett, tovább keres. Ha nem volt név, vagy a név egyezett, beveszi a rekordot. LOAD után a CR-t még azelőtt meg kell nyomni, hogy a magnón a fütty megszólal.
Ha a beolvasás hibás volt, de csak a program szövegében van hiba, akkor ERROR üzenettel tér vissza. Ha máshol volt a hiba, és a beolvasás teljesen használhatatlan, újra bejelentkezik a gép, és ekkor alapállapotba kerül a BASIC-is. A LOAD parancsot lehetőleg a kép tetején adjuk ki, mert az utolsó sorban a soremelés miatt hibás lesz a név kiírása.

MON A$

Ez az utasítás a mögötte álló Stringkifejezésben megadott Monitor vagy Assembler parancsot hajtja végre. A Stringnek pontosan úgy kell kinéznie, mintha a Monitorban adták volna ki. A Monitor-parancs végrehajtása után visszatér a BASIC-be.

N E W

Alapállapotba hozza a BASIC-et. Ez azonos lesz a bekapcsolás utáni állapottal. Ha utána egy aritmetikai kifejezés is áll, akkor ez annak a memóriahelynek a decimális címe lesz, ahol a BASIC-program kezdődni fog.

ON X

Szokásos alakja ON X GOTO 10,20,30 MÁS GÉPEKEN!!!
Az eddig használt listáknál a , elválasztó jel minden listaelem figyelembevételét előirta. Pl. PRINT A,B A/X,Y/ GOSUB 10,20. Viszont ON utasításnál a listának csakis egyetlen /persze X-től függő/ eleme számit. Az ON utáni lista lehet tehát egy exkluziv lista, amely elemeinek elválasztására új jelet érdemes bevezetni.
Ez az aposztrof. Eszerint a helyes formátum a HOMELAB - BASIC-ben ON GOTO' 10'20'30 lesz. /Itt az első listaelem előtt is kell ' /. Az ON után tehát a GOTO utasitás "ki van emelve" és az utána következő exkluziv lista az argumentuma. Ennek analógiájára bármely utasitás -kivéve az értékadást- kiemelhető ON után.
Pl. ON X PRINT' A'B'C

X értéknek megfelelően A,B vagy C értéket nyomtatja ki.

Ha X különböző értékeire különböző utasitásokat kell végrehajtani, tehát az utasitás nem emelhető ki, akkor az ON X után egy exkluziv utasitás-lista is állhat.

```
Pl.  1Ø ON X ' PRINT A : C = 3 ' PRINT D ' FOR J = 1
     TO 7 :U (J) = Ø:NEXT
     2Ø GOSUB 1ØØ ...
```

Ezt a hagyományos módon a következő programrészlet oldja meg:

```
1Ø                ON X GOTO  1ØØØ, 11ØØ, 12ØØ
2Ø                GOSUB 1ØØ...
1ØØØ              PRINT A : C = 2 : GOTO 2Ø
11ØØ              PRINT D : GOTO 2Ø
12ØØ              FOR J = 1 TO 7 : U (J) = Ø : NEXT :GOTO 2Ø
```

N E W

Alapállapotba hozza a BASIC-et. Ez azonos lesz a bekapcsolás utáni állapottal. Ha utána egy aritmetikai kifejezés is áll, akkor ez annak a memóriahelynek a decimális címe lesz, ahol a BASIC-program kezdődni fog.

ON X

Szokásos alakja ON X GOTO 10,20,30 MÁS GÉPEKEN!!!
Az eddig használt listáknál a , elválasztó jel minden listaelem figyelembevételét előirta. Pl. PRINT A,B A/X,Y/ GOSUB 10,20. Viszont ON utasításnál a listának csakis egyetlen /persze X-től függő/ eleme számit. Az ON utáni lista lehet tehát egy exkluziv lista, amely elemeinek elválasztására új jelet érdemes bevezetni.
Ez az aposztrof. Eszerint a helyes formátum a HOMELAB - BASIC-ben ON GOTO' 10'20'30 lesz. /Itt az első listaelem előtt is kell ' /. Az ON után tehát a GOTO utasitás "ki van emelve" és az utána következő exkluziv lista az argumentuma. Ennek analógiájára bármely utasitás -kivéve az értékadást- kiemelhető ON után.
Pl. ON X PRINT' A'B'C

X értéknek megfelelően A,B vagy C értéket nyomtatja ki.

Ha X különböző értékeire különböző utasitásokat kell végrehajtani, tehát az utasitás nem emelhető ki, akkor az ON X után egy exkluziv utasitás-lista is állhat.

```
Pl.  1Ø ON X ' PRINT A : C = 3 ' PRINT D ' FOR J = 1
     TO 7 :U (J) = Ø:NEXT
     2Ø GOSUB 1ØØ ...
```

Ezt a hagyományos módon a következő programrészlet oldja meg:

```
1Ø                ON X GOTO  1ØØØ, 11ØØ, 12ØØ
2Ø                GOSUB 1ØØ...
1ØØØ              PRINT A : C = 2 : GOTO 2Ø
11ØØ              PRINT D : GOTO 2Ø
12ØØ              FOR J = 1 TO 7 : U (J) = Ø : NEXT :GOTO 2Ø
```

Egyetlen kikötés, hogy az ON utáni utasitás -lista nem tartalmazhat újabb ON utasitást, vagy '-os IF-et. /Lásd később/.
Aposztrofhoz érve a program-végrehajtás abbamarad, és a következő sornál folytatódik. Az előző példában tehát a program minden X mellett a 20. sorban folytatódik. Mivel a programsor hossza nincs korlátozva, az ' bevezetésével az ON utasitás határtalan lehetőségeket nyit a programozó számára. Komplett eljárások irhatók meg egyetlen sorban.

IF

A relációkról elmondottak alapján nem meglepő, hogy az IF tulajdonképpen az ON kétértékű megfelelője. Hogy a BASIC kompatibilitása megmaradjon, a THEN szócskát megtartottuk, bár jelentése itt megegyezik az ' . További ' segitségével az ELSE funkciója valósitható meg. /Ekkor a szigorúan vett ON-ban a reláció Ø értéke 2 kellene legyen. Ez az egyetlen különbség./

Az alábbiakban példák láthatók helyesen felirt IF utasitásokra:

```
IF    A  <   D    THEN    A=-A : B = A
IF    A  >   D    THEN    A= Q ' B = P
IF    A  >=  D    PRINT   'A ' B
IF    A  >   D    GOTO    '10'100
IF    A  ><  D    PRINT   'A:C = 5  'B:E = 6+B
IF    A   =  D    'B = 0 : PRINT Z ' S = 4 : GOTO 100
```

PLOT X,Y

A CR változó által meghatározott szinü pontot teszi ki a képernyő /x,y/ koordinátájú pontjára. A / Ø,Ø / pont a bal alsó sarokban van. X a vizszintes, y a függőleges koordináta. Az utasitás CR értékét nem változtatja meg. A koordinátákra nézve: $Ø \leq X \leq 127$ vagy 63; $Ø \leq Y < 95$

POKE I,J..

J értékét az I cimü memóriahelyre teszi. I-re ugyanaz érvényes, mint PEEK (I) nél, azzal a különbséggel, hogy itt, ha $I < 8192$, akkor az adott cimmel egy OUT /gépikódú/ utasitást hajt végre. /Output perifériák kezelhetők./ J után,-vel elválasztva további kifejezések is irhatók. Ezeket a soronkövetkező cimekre irja be.

Ha I nagyobb mint 65536, akkor itt is a második lap címei érvényesek.

POP

Ha FOR ciklusból, vagy BASIC szubrutinból NEXT vagy RETURN utasitás nélkül akarunk kilépni, akkor ezt POP-utasitás után tehetjük meg. A ciklus és a szubrutin eltárol egy visszatérési címet a stack memóriába. NEXT és RETURN utasitásnál ezeket a gép mindig előveszi a stackből. Ha tehát többször kilépünk NEXT vagy RETURN nélkül, a stack lassanként megtelik. Az is előfordulhatna hogy egy -félbehagyott ciklus keletkezik, és egy későbbi NEXT nem azt a ciklust zárja le, amit kellene.
Ezt akadályozza meg a POP utasitás.

```
Pl:  1Ø FOR J = 1 TO V : IF A (J) = c (k) THEN POP:
        GOTO  1ØØ  '  NEXT
```

PRINT

Kinyomtató utasitás. A kulcsszó után a nyomtatási lista áll. Ez kifejezésekből, változókból, a CUR-ból, szám ill. stringkonstansokból állhat. A nyomtatási lista elemei között vessző és pontosvessző az elválasztójel. Ezek egyben a nyomtatási formátumot is meghatározzák. A pontosvessző a listaelemek egyszerű egymás után irását eredményezi. A vessző a következő zóna elejére állitja a kirót. Ebben a gépben 1 sor 4 zónából áll, amelyek egyenként 8 vagy 16 karaktert tartalmaznak. Ha a listaelem után nem áll semmi, az a kiirás után egy soremelést eredményez.
Üres PRINT utasitás egy sort emel. A PRINT kulcsszót a kérdőjel teljes mértékben helyettesiti, elég tehát csak azt irni PRINT helyett.

PRINT # X;

Az OUTPUT perifériák kezelésére szolgáló utasitás. Az X szám vagy kifejezés, az Output-eszköz száma. # Ø a TV-display-t jelenti, mig a # 1 a PRINTER-t. A további output eszközök kezelését az INPUT-hoz hasonlóan szintén külön kell megirni. Ebben az OUTPUT # VEKTOR-ról irottak
A CUR hatása nem jelenik meg a printeren, a CUR mindig a dispay-en érvényes.

READ X1, X2

A DATA utasitás párja. READ hatására X1, X2 stb. változók rendre felveszik a DATA utasitások után felsorolt értékeket. Ezt úgy lehet szemléltetni, hogy az összes DATA utáni adatokat sorban beteszük egy táblázatba és a READ utasitással egyenként olvassuk ki őket. A READ-nak van egy "mutatója", amit minden értékadás után eggyel tovább állit. Ujabb READ utasitás-onnan folytatja a táblázat olvasását, ahol az előzőt abbahagyta.
Ha a DATA után álló kifejezés hibás, akkor a hibaüzenet a READ sorában keletkezik.

Ha kevesebb az adat, mint a READ, akkor OD Error-t uzen.

REM

A REM után megjegyzések helyezhetők el. Végrehajtáskor a gép ezeket a szövegeket egyszerüen átugorja.

RESTORE

A DATA-val definiált adatmező elejére állitja a READ "olvasó-mutatóját". Ha a RESTORE után egy kifejezés áll, akkor azt kiszámolja, és a READ "olvasó-mutatóját" ennek a sornak az első DATA-jára állitja. Ha nincs DATA a sorban, akkor az ezután következő első DATA-ra áll.

RETURN

Visszatérés szubrutinból, GOSUB-utasitás párja.
GOSUB nélküli RETURN esetén P_p Error keletkezik.

RUN

Elinditja a programot. Mielőtt elkezdené végrehajtani a programot, először törli a változótáblát, és csak utána tér rá a legkisebb számu sorra. Ha RUN után egy szám áll, akkor a végrehajtás az adott számnál kezdődik. Hatása parancs módban sem azonos a GOTO-val, mert a GOTO nem csinál változó törlést. Pl. egy programhiba miatt leállt program tovább futtatható egy másik sortól a GOTO segitségével. RUN esetén elvesznének a változók értékei.

SAVE "NÉV"

Program eltárolása kazettás magnóra. Idézőjelek között a rekord nevét kell beirni. CR megnyomása előtt a magnót kell elinditani felvétel állásban. Tárolás alatt a gép fütyül. A tárolás $4Ø16-tól a Basic Text végéig tart.

(LET)

Ilyen utasitás sincs ebben a BASIC-ben, az értékadás egyszerüen A = x alakú, ahol A egy változó, x pedig egy kifejezés. A LET szó nem is szerepel a kulcsszavak között.

ÜZENETEK - HIBAÜZENETEK

A gép bekapcsolásakor, ill. minden RESET alkalmával bejelentkezik. Ha a BASIC nem müködőképes, akkor inicializálódik. /Pl. bekapcsoláskor, ill. olyan programhiba esetén, ami a rendszerváltozókat megváltoztatta/. Ekkor kiirja a szabad memóriaterületet, és alapállapotba hozza a BASIC-et. Ha RESET-kor a BASIC "életképes" volt, ez nem történik meg. Ekkor a program megmarad, és a gép csak OK üzenetet ir.
Minden végrehajtott parancs után OK üzenet keletkezik. BRK-val megszakitott program esetén BREAK üzenetet ad, és kilistázza a sort, ahol a futás megszakadt.
A BASIC hibaüzenetek általános alakja:
- hibás parancs esetén: V ERROR,
- hibás program esetén: V ERROR, és a hibás sor listája.

V itt a hibára jellemző kód, amit alább részletesen ismertetünk.

Programhiba esetén a rossz sor listája is megjelenik, segitve ezzel a hiba javitását. /Ez a kiirt sor a cursorral azonnal kijavitható./

A HIBAKÓDOK A KÖVETKEZŐK:

BS BAD SUBSCRIPT : Méreten kivül eső tömbindex

CN CONTINUE ERROR: Nem folytatható a futás

DD DOUBLE DIMENSIONED VAR: Kétszer dimenzionált változó

IQ ILLEGAL QUANTITY: Az argumentum kivül esik az értelmezési tartományon

OD OUT OF DATA : Több a READ utasitás, mint a DATA

OM OUT OF MEMORY: Elfogyott az engedélyezett memóriaterület

OV OVERFLOW : Túlcsordulás, ábrázolhatatlanul nagy eredmény keletkezett

PP POP ERROR : NEXT FOR nélkül, RETURN GOSUB nélkül vagy POP FOR ill. GOSUB nélkül

SL STRING TOO LONG: 255-nél hosszabb STRING keletkezett.

SN SINTAX ERROR : Szintaktikus hiba

TM TYPE MISMATCH : Az előfordult kifejezés tipusa nem megfelelő

US UNDEFINED STATEMENT : Nem létező sorra történt hivatkozás

/Ø DIVISION BY ZERO : Ø-val történt osztás

Speciális karakterek kódjai

5	BELL	/ harang /	11	→	/cursor jobbra /
6	INS	/ INSERT /	12	CLS	/képernyő törlés/
7	DEL	/ DELLETE /	13	CR	/ CARRAGE RETURN/
8	↓	/cursor le /	14	TAB	/ tabulátor /
9	↑	/cursor fel /	15	HOME	/ Kép elejére /
10	←	/cursor balra /			

A GÉPIKÓDÚ MONITOR

Ezt a gépet elsősorban BASIC programok számára fejlesztették ki, ezért assembler támogatása alapkiépítésben nincs /Bővítésnek megvehető/. A monitor csupán az alapvető funkciókat látja el: memóriaterületet listáz, memóriába adatot tölt, memóriaterületet kazettára kitárol és a programot a megadott címtől elindítja. A monitor BASIC-ből CALL 892 /decimális/ hívható be. A monitor új sort kezdve jelzi, hogy a felhasználó rendelkezésére áll. Ezután gépelhetők be az egyes parancsok. A monitorban minden szám hexadecimálisan értendő.

MONITOR PARANCSOK

D X Y vagy #D X Y

Memóriaterületet listáz X címtől Y címig.
Hogyha a végcím (Y) hiányzik, akkor a D hatására -a képmérettől függően- 128 vagy 256 bájt íródik ki, soronként 8 vagy 16 bájt. A sorok elején a kettőspont után a sor első bájtjának címe van. Ha D után nem áll szám, onnan folytatódik a kiírás, ahol legutóbb abbamaradt. A Ø kezdőcímű listázás nem lehetséges, akkor a gép nem veszi figyelembe az X címet.
Ha a D előtt # áll, akkor az előbbi szabályoknak megfelelően a printerre listáz.

:X P Q R ...

Memóriaterületet tölt be. X az a cím, ahová az első számot / P / teszi. A Q R ... stb. számok egymás után a soronkövetkező címekre kerülnek be. A CR megnyomásakor mindig kiírja a következő címet és új adatot vár. Üres sorra visszatér a monitorba.
A cím és az első adat között egy betűköz áll. További betűköz már nem szükséges, a számok sorban, kétjegyenként töltődnek be. Idegen /nem space vagy hex szám/ karakterre a betöltés abbamarad. Mivel D a lista elé kettőspontot ír, ezért az általa listázott sorok cursor-ral javíthatók. CR-re a javított sor töltődik be.

G X

Programot indit el az X cimtől.
A program végén, ha csak valami különös ok nincs, a monitorba célszerü visszatérni egy RET utasitással.

S X Y Z

Kitárolja kazettás magnetofonra az X-től Y-ig terjedő memóriaterületet. A rekord neve Z lesz. A név a második címet követő első nem space karaktertől a sor végéig tart. Kitárolás közben a beépitett hangszóróból a magnetofonra felvett hang hallható. A monitorból kitárolt rekord formátuma azonos a BASIC-évei, ezért a beolvasás mindig BASIC-ben történik.

Az itt felsoroltaktól eltérő parancs-betükre a gép hibát üzen.

RESET

A készülék bekapcsolás után BASIC-ben ébred. Előtte memória tesztet végez és beállitja a BASIC rendszerváltozóit. Ezt a folyamatot inicializálásnak nevezzük. Ha a gép valamilyen okból végtelen gépikódú ciklusba kerül, innen a RESET-gombbal térithetjük vissza. Ez az NMI felhasználásával szoftver közbeiktatásával ugrik a Ø cimre. Ez a RESET újra készit memória tesztet, de e közben a memóriatartalom nem változik, tehát programot nem töröl. A teszt után megnézi, hogy a BASIC "életképes"-e még. Ha minden rendben van, beugrik a BASIC-be és folytatódhat a programozás.
Ha valamilyen okból a rendszerváltozók átiródtak, a BASIC újra inicializálódik. Ekkor úgy jelentkezik be, mint bekapcsoláskor, és igy a BASIC program is elveszik.
Pereze a gépikódú rész még igy is megmaradhat, hiszen azt a BASIC inicializálódás nem érinti.

A RESET kapcsán van még egy emlitésre méltó dolog. Nevezetesen az, hogy a BASIC-be ugrás előtt a gép megnézi, hogy a 2ØØØ /Hex/ cimen Ø van-e. Ha az van, nem megy a BASIC-be, hanem a 2ØØ1 cimre ugrik. Ezáltal mód van a software-rendszer teljes módositására, ROM-ban tárolt programok közvetlen behivására.
Mivel a gépben van hely ROM-bővitésre, az előbb emlitett tulajdonság minden külső csatlakozó vagy bővités nélkül felhasználható.

KAZETTÁS ADATRÖGZITÉS /HANGKELTÉS/

Mivel mind az irást, mind az olvasást szoftverrel oldottuk meg, lehetőség van olyan programok /saver és loader/ készitésére, amelyek más gép számára készitenek kazettát, ill. más gépek kazettáit tudják elolvasni.

Ez adott esetben hasznos lehet, de közvetlen programcseré̋t sajnos nem lehet igy végezni. Legalábbis BASIC szinten nem! Mindenesetre kis ügyességgel adatok, szövegek vagy gépikódú programok igy átvihetők egyik gépről a másikra.

Ahhoz, hogy ilyen saver-loader programokat irjon valaki, már alaposan tisztában kell lennie mindkét gép "lelkivilágával". Ezt elősegitendő, most leirjuk a gép tárolórendszerét.

Először lássuk a rekord formátumát!
Minden rekord egy kb. 2 másodperces headerrel kezdődik. Ez a szinkornizálást szolgálja. A header csupa LO bit-et tartalmaz. A headert egy promt-byte /A5/ zárja. Ez után a rekord neve jön, amit egy NULL karakter / ØØ / zár. Ezt az eltárolt terület kezdőcime és hossza követi. Ez után jönnek az adatok és végül egy ellenőrző bájt. Ez a bájt az adatok összegének legkisebb helyiértékü bájtja. A rekord legvégén jön egy blockzáró karakter. Ha ez nem nulla, akkor megint jön egy block, amelynek headerje rövidebb, és nincs neve, de egyébként az előzővel azonos szerkezetü. Ha Ø a záró-karakter, akkor befejeződik a rekord.

header	A5 prom	név	ØØ	SL start LO	SH start HI	LL length LO	LH length HI	data	CH check byte	BE block end

A kiadott adat bájtok között nincs szünet, sem paritásbit, a rekord tehát egy bit-folyam. A magnetofonra kitárolt jelalak a következő.

Tehát egy bit akkor 1, ha két szinkronozó jel között középen van egy harmadik impulzus is. A magnetofon ezt a jelalakot nem tökéletesen viszi át, de a jelölt idők, a nyávogástól eltekintve állandók.

Ilyen -tárolásra alkalmas- jeleket hangbit föl-le kapcsolgatásával /a keyboardmező ØØ és 8Ø címének megszólításával/ lehet előállitani. Ez egyébként vonatkozik mindenfajta hangkeltésre is.

Ezzel a megoldással tehát tetszőleges soros formátum előállitható, csak pontosan ki kell számolni a program futási időit.

A beolvasás szintén elég egyszerűen történik. Ha a keyboardmező negyedik címét olvassa a Ø.biten a soros bemenetet kapja. Nincs más dolga, mint megszámolni, milyen hosszú ideig van itt LO ill. HI. Ebből már egy egyszerű összehasonlitással eldönthető, hogy a bit HI vagy LO. Más formátumú kazetták ugyanezen az elven olvashatók, csak legföljebb nem ilyen egyszerű a dekódolásuk.

A bit formátum megváltoztatásán kivül gondoskodni kell még arról is, hogy a rekord formátuma is megegyezzen, ehhez azonban alaposan kell ismerni a kérdéses gépet. A HOMELAB 3. gépkazettás rendszere egyébként teljesen azonos a korábbi HOMELAB /AIRCOMP 16/ géppel.

INPUT - OUTPUT RUTINOK

RST 1 — A magnokezelés használja

RST 6, RST 7 — Szabad a felhasználó számára. Ezek a restartok egy-egy vektor által mutatott cimre ugranak. Az RST-vektorok helye rendre 4043 403F .

RST 3 — Input egy karakter. A 4002 vektoron keresztül működik. Az A-ba beveszi az éppen lenyomott gomb ASCII kódját. Ha semmi nincs lenyomva, akkor A=Ø. Ha F1 vagy F2 is le van nyomva, az A-ban akkor is csak a gomb kódja lesz, de a gép elugrik a 4006 vektor által mutatott cimre. Hogy F1 vagy F2 volt lenyomva, az a D regiszter 1. bitjéből derül ki. Alapesetben a 4006 vektor olyan programra mutat, ami a grafikus karaktereket aktiválja.

RST 4 — Általános pointer rutin.

A ← (DE) és DE ← (DE)+1

RST 5 Output rutin. Az A-ban levő karaktert kiirja a display-re vagy a printerre.
A Ø-tól F-ig a kontrol-karaktereket adja, 1Ø-től 1F-ig pedig a kontrol-karakterek képét, vagyis a karaktergenerátorban Ø-tól F-ig levőket. Igy tehát a valódi 1Ø-től 1F-ig lévő karaktereket nem lehet ezzel kiiratni.

Ø4Ø4 Az outputot a printerre állitja.

Ø423 Az outputot a display-re állitja.
Figyelem! A BASIC utasitásai mindig visszaállitják az outputot a display-re, tehát ez az átállitás csak gépikódból marad hatásos.

Ø546 Egy sort bevevő rutin.
Inputot vár, mig egy CR-t meg nem nyomtak. Akkor beveszi az adott sor első 63 karakterét a 4Ø6Ø-nál kezdődő inputbufferbe. Ide mutat a DE pointer, és ezt kell használni a kiolvasáskor is.

RST 2 Az 546 rutin meghivása után ez a rutin az input-buffer tartalmát adja át a programnak.
E rutin minden meghivásakor az A-ba kerül az input szöveg kovetkező karaktere.

Az utolsó karakter átadásakor a **Z** flag 1 lesz.

037C A MONITOR kezdőcime.

01A0 Kiirja a DE tartalmát hexadecimálisan.

01A5 Kiirja a A tartalmát hexadecimálisan.

03F0 Az A-ba bevesz egy ASCII kódokkal leirt hex bájtot. A kódsorozatra a DE regiszterpár mutat. Mindig az első két karaktert értékeli ki. Ha nem hex szám karaktert talál, befejezi a számbavételt.
A DE pointert a kiértékeléssel együtt tolja előre.

01DC A HL-be bevesz egy ASCII kódokkal leirt hex számot. Az előbb leirtak érvényesen itt is, kivéve hogy itt a B regiszter mondja meg, hogy hány karaktert kell figyelembe venni a bevételkor. Ha B nagyobb mint 4, akkor csak az utolsó négyet értékeli ki. Ha B=0, akkor végig kiértékel, tehát a karaktersorozat utolsó négy hex szám karakterét veszi be.

0188 Táblázatból szöveget ir ki.
A szövegekben csak az első 128 karakter szerepelhet. A legmagasabb bit az egymást követő szövegek elválasztására szolgál. A szöveg utolsó karakteréhez tehát hozzá kell adni 80 -at.
A táblázatot egy FF karakterrel kell kezdeni, és a HL regiszterpárnak erre a cimre kell mutatni. Ezután jönnek a szövegek, az előbb leirt módon elválasztva. A B regiszter mutatja meg, hogy a táblázat hanyadik szövegét irja ki.
A C regiszterbe 0-t kell betölteni!

03E9 DE pointerrel megkeresi az első nem space karaktert az input szövegben. A DE erre a karakterre fog mutatni. Figyelem! Ezt a rutint csak az input bufferben szabad használni.

0533 Átkapcsolja a memóriát a II. lapra.
Figyelem! Az input , output és hangkeltő rutinok ezt automatikusan visszakapcsolják!
A stack átkapcsoláskor a $ 4023-be kerül és csak 7 mélységben használható.

05BE Visszakapcsolja a memóriát az I. lapra.

0108 Hangkeltés. A H regiszterben a hang hossza, az L regiszterben pedig a hang magassága van. Ez a rutin nem végzi el az átkapcsolást.

00F6 A képszinkronhoz szinronizálja a programot. Addig vár, míg egy TV-kép kiszerkesztése be nem fejeződik. Ha csak e rutin meghivása után fordulunk a VideoRam-hoz, akkor elkerülhető a display-n különben megjelenő "szemetelés".
Ez a rutin sem vált memórialapot, tehát meghivása előtt át kell kapcsolni. /természetesen csak akkor, ha egyáltalán van II. lap a rendszerben./

061F Load Header

0616 Load egy bájt az A-ba

074E Load az első érvényes rekordot

0751 Load egy rekordot.
A névre a DE pointer mutat, és a nevet 00 22 vagy 60 karakter zárja

061C Save header

06C9 Save egy bájtot az A-ból

07A3 Save egy rekordot. A BC-ben a kezdőcim, a HL-ben a végcim van. A névre u.a. érvényes, mint Loadnál.

BASIC RUTINOK

1D4D A BASIC kezdőcime.

18E1 Hangkeltés. Az A tartalmazza a hang magasságát, a C pedig a hosszát.

18E7 Dallamjátszás. /BEEP /
A DE pointer a dallamtábla elejére mutat. A dallamot a 20 karakter zárja le. A táblázatban lévő karakterek 21-től 3F-ig a ritmust, 40 fölött pedig a hangmagasságot határozzák meg.

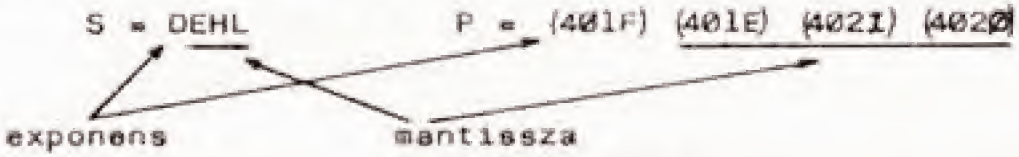
1538 Kiir egy tetszőleges szöveget. Ebben minden karakter szerepelhet. HL mutat a string elejére, E pedig a string hosszát adja.

1DE0 Plot gépikódból.
Az A-ban az Y, az L-ben az X koordináta van. A szint a 404F rendszerváltozó határozza meg. Egyébként a Basic-kel azonos módon működik.

ARITMETIKAI RUTINOK

Az aritmetikai 2 akkumulátort használ: az egyik a P akkumulátor /401E-től 4021-ig terjedő 4 bájt/, a másik az S ami az E D L H regiszterekből áll.

Az akkumulátorok alakja a következő:



Az aritmetika működésének szabályai a Basic-nek megfelelően értendők.

Az általánosan felhasználható aritmetikai rutinok a következők: /a művelet a S és az P között értendő, az eredmény a P-be kerül/.

0718	+	0713	-
07C8	*	06A9	/
0C49	^	0A23	AND
0A18	OR	0C63	>
0C60	<	0C72	><
0C68	=	0C7C	=>
0C77	=<	1098	>=<

A függvények ugyan olyanok mint Basic-ben.
Az argomentum mindkét akkumulátorban szükséges. Az eredmény a P-ben keletkezik.

0974	INT	0994	ABS
0998	SGN	09AA	RND
0B68	COS	0B2C	SIN
0B78	TAN	0AA8	ATN
0C3C	SQR	0BA4	EXP
0BEB	LOG		

Egyéb hasznos aritmetikai rutinok:

063A	P ⟵ HL
0640	P ⟵ NORM (S) : S-et P-be normalizálja
09E6	HL ⟵ INT(S)
0A09	HL ⟵ INT(P) es DE ⟵ INT (S)
0E86	P ⟵ -P
0B8C	P ⟵ NOTP

RENDSZERVÁLTOZÓK

A két bájtos változóknál az alacsonyabb címen található a LO-bájt a következőn pedig a HI-bájt.

4000-01 BASIC, USR vector.
Az USR függvény kezdőcíme: 4000 a LO bájt
4001 a HI bájt

Kezdőérték: 157C

4002-03 INPUT Vector.
Az ide beírt cím lesz az egy karaktert bevevő rutin kezdőcíme.

Kezdőérték: 035C

4004-05 OUTPUT Vector.
Az ide beírt cím lesz az egy karakter kiíró rutin kezdőcíme.

Kezdőérték: 0283

4006-07 F1/F2 VECTOR.
Ha egy gomb mellé az F1 vagy az F2 is le van nyomva, e szerint a vector szerint ugrik el az input rutin folytatni a dekódolást. A kezdőértéknek megadott rutin a grafikus jeleket adja.
Amikor ide ér az input rutin, a A-ban a beütött gomb kódja van, és a Z ill. C statusflag-ek mondják meg, hogy az F1 vagy az F2 volt lenyomva.

Kezdőérték: 0600

4008 SHIFT ALTER flag.
Ez mutatja meg, hogy Shift nélkül nagy vagy kisbetű legyen. Ha páros kisbetű van.

4009 SCROLL stop flag.
Ha ez a flag Ø , akkor a képalján megáll a kiírás és csak a SHIFT/Space hatására megy tovább.

400A 32/64 betü flag.
Ha 1 akkor 64 betüsakiirás, ha 2 akkor 32 betüs, de ehhez hardver átkapcsolás is kell.

400B Buffer-pointer
Ez az output buffer pointere, szövegkiíráshoz használják.

400C-0D RST 1 VECTOR
RST 1 hivásakor e szerint ugrik tovább.

400E-0F INPUT # Vector.
INPUT # 1, ill. ennél nagyobb számú input-perifériák lekezelésére.

402E-2F Current Line
Az aktuális, éppen végrehajtott BASIC sor száma.

4030-31 Start of BASIC Text

Kezdőérték: 40A0

4032 Flags for BASIC
BASIC flegek, a felhasználó számára értéktelen.

4033-34 Auxiliary Stack pointer. A stack pointer átmeneti tárolására szolgál.

4035-36 Statement pointer for CONT. A következő végrehajtandó utasitás cime a RAM-ban.

4037-38 Next intem for READ. A következő érték cime a RAM-ban, amit a READ olvasni fog.

Kezdőérték: 40A0

4039-3A Current Line for CONT. Erre a sorra tér vissza CONT utasitásnál.

403B-3E Random Number last
A legutóbbi 32 bites véletlen bitsorozat.

403F-40 RST 7 VECTOR

4041-42 Free

4043-44 RST 6 VECTOR

4045-46 Free

4047-48 Display pointer for screen-editor
Display pointer az editor müködéséhez.

4049-4A ERROR VECTOR
Ide kell beirni az errort kiszolgáló rutin cimét. Alapesetben a normál hibauzenetre mutat.

Kezdőérték: 1581

404B	INTERRUPT FLAGS
404C	Security flag Védi a programot, ha az értéke 00. Kezdőérték: 10
404F	CR változó. Ez a BASIC CR változója.
4052-53	Vector for Interpreting A BASIC-fordító rutin kezdőcime. Bővitéskor kell, a felhasználónak értéktelen. Kezdőérték: 16B2
4054-55	Vector for Listin A BASIC-listázó rutin kezdőcime. Bővitéskor kell, a felhasználónak értéktelen. Kezdőérték: 0167
4056-57	Vector for Error A BASIC-error üzeneteket generáló rutin kezdőcime. Ezt csak bővitéskor kell módositani, felhasználónak értéktelen. Kezdőérték: 1581
4058-59	Start of token table A BASIC kulcsszókészletét tartalmazó táblázat kezdőcime. Kezdőértéke: 1C3C
405E	PRINTER POS. A printer fejének poziciója.
405F	PRINTER FLAG. A printeléskor használt flag.
4060-9F	OUTPUT és INPUT BUFFER, a kiirás és beolvasás gyüjtője.

404B	INTERRUPT FLAGS
404C	Security flag Védi a programot, ha az értéke 00. Kezdőérték: 10
404F	CR változó. Ez a BASIC CR változója.
4052-53	Vector for Interpreting A BASIC-fordító rutin kezdőcime. Bővitéskor kell, a felhasználónak értéktelen. Kezdőérték: 16B2
4054-55	Vector for Listin A BASIC-listázó rutin kezdőcime. Bővitéskor kell, a felhasználónak értéktelen. Kezdőérték: 0167
4056-57	Vector for Error A BASIC-error üzeneteket generáló rutin kezdőcime. Ezt csak bővitéskor kell módositani, felhasználónak értéktelen. Kezdőérték: 1581
4058-59	Start of token table A BASIC kulcsszókészletét tartalmazó táblázat kezdőcime. Kezdőértéke: 1C3C
405E	PRINTER POS. A printer fejének poziciója.
405F	PRINTER FLAG. A printeléskor használt flag.
4060-9F	OUTPUT és INPUT BUFFER, a kiirás és beolvasás gyüjtője.

4000	-	4001	USR vector
4002	-	4003	RST 3 input vector
4004	-	4005	RST 5 output vector
4006	-	4007	F1 / F2 vector
4008			SHIFT ALTER flag
4009			Stop Flag for scroll
400A			32/64 karakter flag.
400B			buffer-pointer
400C	-	400D	RST 1 vector
400E	-	400F	INPUT # vector
4010	-	4011	PRINT # vector
4012	-	4013	Monitor-pointer
4014	-	4015	Cursor-pointer
4016	-	4017	Highest Memóry pointer
4018	-	4019	End of Basic text
401A	-	401B	End of variable table
401C	-	401D	End of string-space 1
401E	-	4021	Primery accumulator
4022	-	402B	Auxilary stack
402C	-	402D	End of string-space 2
402E	-	402F	Current line
4030	-	4031	Stat of Basic Text
4032			Basic Flags
4033	-	4034	Auxilary stack pointer
4035	-	4036	Statement pointer for CONT
4037	-	4038	READ-pointer
4039	-	403A	Current line for CONT
4038	-	403E	Last Random Number
403F	-	4040	RST 7 vector
4041	-	4042	Free
4043	-	4044	RST 6 vector
4045	-	4046	Free

B Ő V I T É S E K

A következő oldalon a HOMELAB III. személyi számitógéphez vásárolható bővitések használatának leirása található.

Ezek a kovetkezők:

Bővitett BASIC	$ 2ØØØ
ASSEMBLER	$ 28ØØ
Duplapontos Aritmetika	$ 3ØØØ

A félreértések elkerülése végett ezeket a bővitéseket külön kell megvásárolni a géphez.

BŐVITETT BASIC

A Bővitett Basic elnevezésü program egy 2K-s toldalék az alap Basic-hez. A szükös 8K-t 10K-ra bővitve számos olyan új lehetőség valósulhatott meg, ami a használat során sokaknak hiányzott.

A Basic Bővités használata rendkivül egyszerü. Csak be kell helyezni a beégetett IC-t /közvetlenül a Basic mögé/ és bekapcsoláskor automatikusan aktivizálódik a bővités. Ennek első látható jele, hogy a gép magyar nyelvü hibaüzeneteket küld. /A Basic többi része megmarad angol nyelvünek. Az utasitás-készlet magyarra forditása kimaradt, mert ezt a legtöbben fölöslegesnek tartották/.

És most lássuk az új utasitásokat, függvényeket!

UTASITÁSOK

DELETE	EDIT	ELOAD
ESAVE	GOSUB#	GOTO#
KEY	MERGE	REPEAT
UNTIL	VERIFY	

B Ő V I T É S E K

A következő oldalon a HOMELAB III. személyi számitógéphez vásárolható bővitések használatának leirása található.

Ezek a kovetkezők:

Bővitett BASIC	$ 2ØØØ
ASSEMBLER	$ 28ØØ
Duplapontos Aritmetika	$ 3ØØØ

A félreértések elkerülése végett ezeket a bővitéseket külön kell megvásárolni a géphez.

BŐVITETT BASIC

A Bővitett Basic elnevezésü program egy 2K-s toldalék az alap Basic-hez. A szükös 8K-t 10K-ra bővitve számos olyan új lehetőség valósulhatott meg, ami a használat során sokaknak hiányzott.

A Basic Bővités használata rendkivül egyszerü. Csak be kell helyezni a beégetett IC-t /közvetlenül a Basic mögé/ és bekapcsoláskor automatikusan aktivizálódik a bővités. Ennek első látható jele, hogy a gép magyar nyelvü hibaüzeneteket küld. /A Basic többi része megmarad angol nyelvünek. Az utasitás-készlet magyarra forditása kimaradt, mert ezt a legtöbben fölöslegesnek tartották/.

És most lássuk az új utasitásokat, függvényeket!

UTASITÁSOK

DELETE	EDIT	ELOAD
ESAVE	GOSUB#	GOTO#
KEY	MERGE	REPEAT
UNTIL	VERIFY	

FÜGGVÉNYEK

DEC	HEX$	FORM$
FRA	FSW	MAX
MIN	MOD	ROUND
STRING$	VAR	

RENDSZERVÁLTOZÓ

PRG

SPECIÁLIS KARAKTEREK

% $

Bevezettünk egy új rendszerváltozót, a PRG-t. Ez csak olvasható változó. Azt adja meg, hogy a lehetséges Basic lapok /lásd EDIT/ közül éppen melyikben vagyunk.

Van két új speciális karakter:

% a mögötte álló számkonstanst binárisan értelmezi. A % mogott ezért csak Ø vagy 1 állhat. A bináris számkonstans hossza legfoljebb 16 jegy lehet.

Pl.: A=%1ØØ11 /A=19/

$ a mögötte álló számkonstanst hexadecimálisan értelmezi. A $ mogött ezért csak Ø ...9 és A B C D E F állhat. A $ után álló jelsorozatból a gép az utolsó 4 jegyet értelmezi.

Pl.: A= $3F39 /A=16185/

UTASITÁSOK

DELETE A-B

A kulcsszó után egy címintervallum áll. A címintervallum a LIST-tel azonos módon értendő.
A DELETE a címintervallum által meghatározott programsorokat kitorli. Ha a DELETE után nem áll semmi, az összes sort törli.

A gépben egyszerre több -egymástól független- Basic program elhelyezése lehetsége, és ennek megfelelően parancsmód is több lehet. Ezeket Basic lapoknak nevezzük. Az EDIT N ezek között biztosit átmenetet.

Alapesetben a gép a Ø. lapon van. Tegyük fel, hogy oda már irtunk egy programot. Ha most kiadjuk az EDIT 1 parancsot, a gép kitörli a változótáblát és átlép egy másik lapra. Ott új, az előzőtól független program irható. /Tehát újra lehet használni ugyanazokat a sorokat/. Ha itt listázunk, vagy itt adunk RUN parancsnot, akkor az csak ezen a lapon lévő programra lesz érvényes.

EDIT 2-vel újabb lapot nyithatunk, mig EDIT Ø-val visszatérhetünk a Ø lapra /alaplap/.
Tehát az EDIT N parancsmódban átteszi a vezérlést az N. lapra. Ha már korábban megnyitottuk ezt a lapot, akkor nem töröl változótáblát. Ha ilyen számú lap még nem volt, akkor megnyitja a következő lapot, és törli a változótáblát. /Figyelem! ha Pl. 5 lap van nyitva, és EDIT 8-at adunk, akkor új lapot nyit, de az 6. lesz/.

Megnyitott lapok megszüntetése nem lehetséges. Ha egy lap szükségtelen, egyszerüen csak törölni kell belőle a programot. Hogy éppen melyik lapon dolgozik a gép, azt a PRG rendszerváltozó mutatja meg. Az alaplapon a PRG=Ø.

A különböző lapokon elhelyezett programok közos változótáblával dolgoznak, de a programok egymástól függetlenek, és egyszerre csak az egyik lapon futhat program.
Az egyik lapon futó program átugorhat egy másik lapra, vagy meghivhat egy másik lapon lévő szubrutint. Ezekre külön utasitások szolgálnak.

Magnóra tároláskor az összes megnyitott lap eltárolódik függetlenül attól, hogy melyik lapon adtuk ki a SAVE parancsot. Ugyanigy a LOAD is beveszi az összes kitárolt lapot. d4
Ha az EDIT után nem áll szám, akkor az az EDIT Ø-val azonos. A programban kiadott EDIT N megállitja a programot és a gép parancsmódba megy az N. lapon.

GOTO # N,A

Ez az utasitás az N. lapon lévő program A. sorára ugrik. N is és A is lehet kifejezés az alap-Basic GOTO-jánál megismert szabályok szerint.

Pl.: GOTO PRG +2,1ØØØ relativan két lapváltás és ott
GOTO 1ØØØ.

GOSUB # N,A

Ez az utasitás az N. lapon lévő program A. soránál lévő szubrutint hivja. N és A ugyanaz, mint a GOTO-nál.

A szubrutin végén, a RETURN utasitás hatására a futás visszatér az eredeti lap megfelelő sorába.

ELOAD " NÉV "	ESAVE " NÉV "
ELOAD X,"NÉV "	ESAVE X," NÉV "

Az első formának azonos a hatása a korábbi SAVE ill. LOAD utasitással.
A második forma különböző perifériákra történő tárolást ill. olvasást jelent. /X=1 a magnót adja/.
Ezekkel az utasitásokkal kell majd például a Bar-Code programokat beolvasni, vagy egy lényegesen gyorsabb magnós tárolást megvalósitani.

```
VERIFY " NÉV "
VERIFY X," NÉV "
```

Ezzel az utasitással a kazettára felvett programokat lehet ellenörizni. /Gépi kódú felvételt is!/.

A VERIFY a LOAD-hoz hasonlóan müködik. "Elolvassa" a kazettát, de nem tölti be a memóriába, csak összehasonlitja azzal. Ha nem talál eltérést, OK üzenettel tér vissza. Ha volt hiba, "Értelmetlen" hibaüzenetet ad.

X ugyanaz mint ELOAD-nál

A VERIFY után a név hiányozhat, akkor az éppen következő programot ellenörzi.

```
MERGE " NÉV "
MERGE X," NÉV "
```

Uj lapot nyit és a kazettán eltárolt programot betölti erre a lapra.

EXT

Alap-Basic-ben ez az utasitás HEX 2000-re ugrott. Mivel a bővités erre a helyre került, ez az utasitás a Bővitett Basic-ben HEX 2800-ra ugrik.

Alapkonfigurációban ide az ASSEMBLER kerül.

KEY

A KEY utasitás megváltoztatja a funkciós gombok jelentését. KEY utasitás után a betük és az F1/F2 gombok kombinációjával a Basic legfontosabb kulcsszavai gombnyomásra bevihetők. Ujabb KEY utasitás visszaállitja az eredeti állapotot, ahol a funkciós gombokkal a grafikus karakterek érhetők el.

REPEAT : ..... : UNTIL R

Ez a FOR - NEXT-hez hasonló ciklusszervező utasitáspár. A REPEAT és UNTIL kozotti programrészletet /ciklusmag/ addig ismétli, amig az UNTIL után megadott R reláció értéke hamis.

Pl.: 10 A=0
20 REPEAT : PRINT A : A=A+1 : UNTIL A=10

ez a programrészlet ekvivalens a kovetkezővel:

10 A=0
20 PRINT A : A=A+1 : IF A 10 THEN GOTO 20

/Ez a példa csak formailag mutatja a REPEAT - UNTIL működését. Valódi haszna itt nem látszik/.

FÜGGVÉNYEK

DEC /A$/

Ha A$ tartalma egy HEX szám ASCII-képe, akkor DEC megadja ennek decimális értékét. /A HEX szám kettes komplemens kódban értendő/.
Az AS elején lévő betüközöket a DEC figyelmen kivül hagyja. Ha a HEX szám 4 karakternél hosszabb, csak az utolsó 4 jegyet veszi figyelembe.

Pl.: DEC (" EFE5") = -4123
DEC ("F") = 15

HEX$ /X/

Ez a függvény előállit egy 4 karakteres stringet, ami az X kifejezés értékének hexadecimális ábrázolásu képe lesz

$-2^{15} < x < 2^{16}$

Pl.: HEX$ (4123) = "1Ø1B" HEX$ (123) ="ØØ7B" d7
HEX$ (-4123) = "EFE5" de HEX$ (61413) ="EFE5"

MIN (I1, I2, ... , In)
MAX (I1, I2, ... , In)

Ezek a függvények az argomentumban felsolt kifejezések értékei közül a legkissebbet ill. a legnagyobbat adják.
Pl. MIN (4, 7, 3, 9) = 3

FSW (X,Y,Z)
FSW (X,Y,Z,U)

Az első forma X függvényében Y vagy Z kifejezés értékét adja.

Ha X = Ø akkor az FSW értéke Y lesz
Ha X ≠ Ø akkor az FSW értéke Z lesz

a második formában ugyancsak X függvényében ad értéket:

Ha X < Ø akkor az FSW értéke U lesz
Ha X = Ø akkor az FSW értéke Y lesz
Ha X > Ø akkor az FSW értéke Z lesz

Példa: a következő program az A I tömb legnagyobb elemét keresi meg

```
T = Ø : FOR I = Ø TO 1ØØ
T = FSW (A (T) > A (I) ,I,T)
```

T-ben a legnagyobb értéku elem indexe lesz.

MOD (X,Y)

X,Y ≠ Ø tetszőleges számok. Ekkor
MOD (X,Y) = X-ABS (Y)*INT (X/ABS (Y))
vagyis X "maradékát" adja Y-al osztva.

Pl.: MOD (26,3) = 2

X,Y lehetnek tortek is. Pl. a következő program szöveget számol át radiánból fok, perccé a MOD segitségével. d8

```
1Ø  INPUT " radián :"; R
2Ø  V  = V*18Ø/P;
3Ø  ? INT (R) :  "  fok  ";
4Ø  ? INT (MOD (R*6Ø , 6Ø)) : "  perc"
```

FRA (X)

X törtrészét adja. FRA (X) = X-INT (X)

Pl.: FRA (PI) = .141593
FRA (-PI) = .858407

ROUND (X,Y)

Az X kifejezés értékét az Y. tizedesre kerekiti.

PL.: ROUND (PI,+4) = 3.1416
ROUND (1532,-2) = 1500

VAR (V)

V tetszőleges tipusú változó azonositója.
A VAR függvény megadja, hogy a V változó értéke hol van eltárolva a memóriában.
Ha V még nem kapott értéket, akkor a VAR értéke Ø lesz.
REAL tipusú változó esetén a VAR által adott cimen kezdődő 4 bájt a változó értéke. Ez az Alap-Basic részben leirt formában értendő. String változónál csak 3 bájt értékes. Az első a string hosszát adja, a másik kettő pedig azt mutatja meg, hogy hol kezdődik a string a memóriában.

STRING$ (X)
STRING$ (X,Y)

Ez a függvény egy X hosszúságú stringet ad, ami csupa Y kódú karakterből áll.
Ha Y hiányzik, akkor az X hosszúságú string a betükóz karakterből fog állni.

FORM$ (X,Y,Z)

A Z kifejezés értékének formatált képét adja meg.
X az egészrész, Y pedig a törtrész számjegyeinek számát adja meg. Az egészrészt és a törtrészt a tizedesjel választja el, és egy további karakter a szám előtt az előjel. Igy a FORM$ egy X + Y + 2 hosszúságú stringet ad.
Ha a Z értéke már nem fér el ebben az ábrázolási tartományban, akkor az előjel függvényében csupa + vagy csupa - string keletkezik.

Figyelem! A FORM$ nem kerekit !

A FORM$ szerkezetét az X és az Y paramétereken kivül egy további paraméter is szabályozza. Ez a HEX 4Ø42 cimen lévő rendszerváltozó, ami 6 bitjében a következőket jelenti.

x	x	5	4	3	2	1	Ø

$ 4Ø42

bit

Ø Ha Ø, akkor a + előjelet nem irja ki /helyette betükoz áll/.
Ha 1, akkor a + előjelet is kiirja.

1 Ha Ø, akkor tizedespontot ir.
Ha 1, akkor tizedesvesszőt ir.

2 Ha Ø, akkor az értéktelen nullákat nem irja ki a törtrészben.

3 Ha 1, akkor Ø törtrész esetén kiirja a tizedesjelet, és egy nullát.
Ha Ø, akkor sem tizedesjelet, sem nullát nem ir.
A 2. bit prioritása nagyobb.

4 Ha Ø, akkor a nulla egészrészt nem irja ki /csak egy tizedesjelet/. dlO

Ha 1, akkor egy nullát ir a tizedesjel elé.

5 Ha Ø, akkor az előjelet a bal szélső helyre irja.

Ha 1, akkor az előjelet közvetlenül az egészrész elé irja.

Példa:

POKE%4Ø42, ØØØØ1Ø11 esetén

FORM%/3,3,5)="+ 5,Ø "

POKE%4Ø42, ØØ11ØØØ1 esetén

FORM%(3,3,1/2) =" +Ø.5 "

HIBAÜZENETEK

BS Rossz tömbindex!

CN Nem folytatható!

DD Ujra méretezett tömb!

IO Nem értelmezhető szám!

OD Több READ, mint DATA!

OM Túl kicsi a tár!

OV Túl nagy szám!

PP Stack hiba!

SL Túl hosszú szöveg!

SN Értelmetlen!

TM Rossz adattipus!

US Hivatkozás nemlétező sorra!

/Ø Ø-val való osztás!

gomb	F_1	F_2
Q	LIST	SIN
W	LOAD	COS
E	MERGE	SQR
R	REPEAT	INT
T	UNTIL	MIN
Y	RETURN	MOD
U	READ	VAL
I	INPUT	INKEY
O	BEEP	CHR$
P	POKE	PEEK
Ö	REM	MID$
Ó	END	STRING$
A	RUN	TAN
S	SAVE	ATN
D	VERIFY	RND
F	FOR	ABS
G	NEXT	MAX
H	GOSUB	ROUND
J	RESTORE	LEN
K	PLOT	POINT
L	CUR	STR$
Ő	CALL	LFT$
Z	DELETE	EXP
X	EDIT	LOG
C	CONT	USR
V	STEP	SGN
B	POP	FSW
N	GOTO	FRA
M	DATA	ASC
Á	DIM	FORM$
É	EXT	RGH$

A bővitett monitor és assembler alapvető fontosságú programcsomag azok számára, akik a Basic programozás mellett gépikódban is hozzá akarnak férni gépukhoz. Tekintettel arra, hogy a gépikódú programozás kb. 10-1000-szeres sebességnövekedést jelent a Basic-hez képest, számtalan esetben elengedhetetlen, hogy a teljes programot, vagy annak kritikus részeit, gépikódú programmal valósitsák meg.

A második szempont, hogy gépikódban alakitható ki a feladathoz illeszkedő optimális adatstruktúra is, ami szintén a sebesség novekedését, de még inkább a felhasznált memória csökkenését eredményezi.

A harmadik terület, ahol gépikódot kell alkalmazni a perifériák és egyéb külső eszkozök szoftverje. Itt szintén a sebességi előny dominál, illetve az, hogy az Interrupt-kérések "azonnal" lekezelhetők, ellentétben a Basic-kel, ahol ki kell várni az aktuális Basic utasitás végét, ami adott esetben több ms is lehet.
Végül meg kell emliteni, hogy a rendszerprogramok nyelve is a gépikód, a gép szoftver lehetőségeinek kibővitésére is ez az egyetlen út kinálkozik.

Tehát mindazok, akik elérték a Basic korlátait, és tovább szeretnének lépni, feltétlenül a gépikód felé kell, hogy forduljanak.

A gépikódú programozás rejtelmeivel itt természetesen nem foglalkozhatunk, erre vonatkozóan jónéhány szakkönyv eligazit. Ugyanigy nem térhetünk ki a Z80 mikroprocesszor utasitáskészletének ismertetésére sem. Ezeket mint ismertnek tételezzük fel, tehát aki nem tud valamit, itt álljon meg és nézze át a szakirodalmat.

A Bővitett Monitor és Assembler /röviden BMA/ 2 K területet foglal el. Megvásárolható ROM-rezidens formában. A ROM-rezidens forma értelemszerüen gépbeépitve mükodik. A BMA-t a Basic-ből lehet hivni EXT paranccsal.

Ugyancsak érvényes a BMA-ra a Basic MON parancsa is. A BMA bejelentkezése után a gép új sort kezd. Ez, és a villogó cursor jelenti, hogy a gép parancsra vár. A parancsok itt egybetusek és ezek után áll az adott parancshoz szükséges további szám, illetve szöveg.

A gép minden parancsot a CR megnyomására hajt végre. A Screen-editor itt is mükodik. A Basic-kel azonos módon itt is bármi bárhol átirható a képernyön és a CR hatására az aktuális logikai sor értelmeződik./Aktuális az a szövegsor, amelyikbe a cursor éppen van/.

Előljáróban talán még csak annyit, hogy a BMA-ban minden szám hexadecimálisan értendő. Ez 16-os számrendszer ahol a 10, 11, 12, 13, 14, 15 jeleket rendre az A, B, C, D, E, F betük reprezentálják.

MOST LÁSSUK A PARANCSOKAT:

D * HEX. formában kiir egy memóriaterületet a képernyőre.

: Memóriatartalmat lehet beirni Hex. formában.

S Memóriaterületet lehet kitárolni kazettára.

G Programot indit el.

F Egy adott területet feltölt egy megadott karakterrel

M Egy memóriaterületet átmásol egy megadott helyre.

C Két memóriaterületet összehasonlit.

T Assembly formában kiir egy memóriaterületet. /visszafordító/

; Gépikódú programot lehet beirni Assembly formában. /odafordító/

R Visszatérés a főprogramba.

Ha olyan parancs előtt # áll, melynek van ontputja, akkor az a printerre fog vonatkozni.
Ilyenek a D, a C, a T parancsok.

E rovid áttekintés után lássuk az egyes funkciókat részletesebben is. X,Y és Z hex számokat fog jelenteni.

F X Y Z

X kezdőcimtól Y-ig feltölti a memóriát/még Y-t is/Z-vel. Z tehát egy kétjegyü szám. Ha ennél tobbet irunk, csak az utolsó két jegyet értelmezi és tölti be.

M X Y Z

X-től Y-ig /Y-t is!/ terjedő részt átmásolja Z helyre. Átmásolás után Z tartalma ugyanaz lesz, mint ami az X-é, Z + 1-é, min X +1-é, stb.
Z-re nincs semmilyen megkötés, bárhova át lehet rakni. /A BMA "megnézi" Z viszonyát X-hez ill. Y-hoz és ennek megfelelően felülről lefelé vagy alulról fölfelé haladva másol. Ez biztositja, hogy másolás közben az eredeti tartalom nem vész el/.

C X Y Z vagy #C X Y Z

Az X-től Y-ig terjedő területet /Y-t is!/ összehasonlitja a Z-nél kezdődő ugyanilyen hosszú memóriaterülettel. Az eltérő tartalmú cimeket felsoroija tartalmukkal együtt. Az első /négyjegyu/szám cimet jelent, a második az azon található adatot. A harmadik az a szám, ami a Z-től kezdődő területben a megfelelő helyen áll. A kulonbségek felsorolását a BMA zónánként és automatikusan végzi el. Az összehasonlitás a jelzett terület végén, vagy az X-gomb megnyomásakor fejeződik be.
A kulönbségek kiirását természetesen a Shift/space-val is meg lehet állitani, ugyanúgy, mint a Basic-ben.

E parancs hatására egy RET utasitás hajtódik végre, tehát a BMA visszatér az őt szubrutinként meghivó programba. Mivel a BASIC-ből is szubrutinként hivjuk a BMA-t, ezért a R parancs alkalmas a BASIC-be való belépésre, feltéve persze, hogy még megvan a visszatérési cim.

Ellenkező esetben G Ø vagy G 1D4D
paranccsal, vagy egyszerüen csak RESET-tel lehet visszatérni.

T X Y vagy # T X Y

X-től Y-ig terjedő részt Z-80 mnemónikus kódba forditja és kiirja. X-re és Y-ra pontosan ugyanaz vonatkozik, mint a D parancsnál.

T a lista elé pontosvesszőt ir, utána jön az utasitás cime, majd a mnemónikus kód.
Ez a mnemónikus kód néhány kivételtől eltekintve a Zilog jelölésével azonos. A kivételek a következők:

Zilog standard	HOMELAB BMA
ADC HL, regiszterpár	ADC regiszterpár
ADC regiszter, szám	ADC regiszter, szám
ADD A, /HL/	ADD /HL/
SBC /IX+d/, /IY+d/	SBC /IX+d/, /IY+d/
BIT Ø, regiszter	BITØ regiszter
BIT 1, regiszter	BIT1 regiszter
. . .	. .
. . .	. .
SET 7, regiszter	SET7 regiszter
CALL C, cim	CALLC cim
. . .	. .
. . .	. .
CALL Z, cim	CALLZ cim

feltételes JP, JR és RET utasitásoknál ugyanigy.

IM Ø	IMØ
IM 1	IM1
IM 2	IM2

IN A, /n/	IN /n/
IN reg /c/	IN regiszter

OUT-nál ugyanigy

OTIR	OUTIR
OTDR	OUTDR

RETN	RETR

A két byte-ot mozgató, vagy abszolut cimre hivatkozó LD utasitások helyett MV használatos. /Az LD a szimpla regiszterek kozti átvitelre van korlátozva.

; X MNEMONIC

Ez a parancs a kettősponthoz hasonlóan mükodik. X cimre berakja annak az utasitásnak a kódját és operandusait, amit utána mnemónikus formában beirtak.
A mnemónikus kódokkal kapcsolatban ugyanaz érvényes, mint T-nél.
A relativ ugrásoknál /JR/ be lehet irni relativ cimet, vagy azt az abszolut cimet, ahova az ugrás szükséges. Ha cimnek FF-nél kisebb szám áll a relativ, ha ennél nagyobb, akkor az abszolut cim lesz.
Formailag a visszafordító úgy mükodik, hogy a beirt sort a CR megnyomása után letörli és visszairja a Standard formátumot. Ez kettős célt szolgál:
Mint majd látni fogjuk, a felhasználónak nem kell mindig mindent precizen beirni az assembly formából, igy az esetleges beirási mód helyett mindig egy tiszta, áttekinthető formátum látszik. Másrészt azonnal meg lehet győződni arról, hogy a gép valóban azt az utasitást vette be, amit szerettünk volna.

Ha a gép nem tudta értelmezni a beirt szöveget nem történik meg a felülirás. A Cursor a helyén marad és újra kell irni az utasitást.
Minden sor bevétele után a BMA pontosvesszőt tesz, és kiirja a soron következő cimet. Ide újabb utasitást lehet beirni, vagy üres sorral alaphelyzetbe visszatérni.

A D-: párhoz hasonlóan módositani lehet a T-által kiirt listát. A cursorral visszalépkedve bármi -cim is- átjavitható, CR-re újra tölthető.
Figyelem! Ha a javitás során a kezdő-cimek megváltoznak ügyelni kell, hogy az abszolut és relativ ugrások helyesek maradjanak. A gép nem végzi el a cimek módositását!
Mint már emlitettük a felhasználónak nem kell beirni azt a teljes formátumot, amit a gép kiir.

Első könnyités, hogy egy komplett utasitás után bármi állhat, az nem zavarja a visszafordităst. Átjavitott soroknál tehát szükségtelen letorolni a sor hátralévő részét, azt a gép már figyelmen kivül hagyja.
Ugyanigy nem kell ügyelni az utasitás operandusai közötti betüközökre, vagy a beirt utasitás helyére a soron belül.

Második, hogy a számot jelölő $-jel elmaradhat, ha egyértelmu, hogy az adott helyen csak szám állhat, /pl. ugrások/. Ahol regiszter is állhat ott ki kell tenni a $-t, mert a HEX számok jegyei egyben regisztereket is jelölnek. A végzárójel helyett lehet betüközt irni, a kezdő zárójel pedig csak akkor szükséges, ha az egyértelmüség azt megkivánja.

DUPLAPONTOS ARITMETIKA

A HOMELAB BASIC számábrázolási pontossága 6 számjegy. Ez sok esetben kevésnek bizonyul. Ennél lényegesen többre van szükség statisztikai, illetve adatfeldolgozási feladatoknál és olyan matematikai eljárásoknál, ahol a hibák összegződnek. Ezért kifejlesztettünk egy olyan aritmetikai bővitést, amely a számkijelzés pontosságát 14 számjegyre növeli.

A SZÁMÁBRÁZOLÁSRÓL

A duplapontos csomagban minden szám 8 byte-os lebegőpontos alakban van eltárolva. Ez egyben azt is jelenti, hogy a belső alapmüveletek relativ pontossága 2E-17, mig az ábrázolható számok tartománya 1E-32-től 1E+32-ig terjed. Függvények esetén ez a relativ hiba valamivel nagyobb, de mindenképpen 1E-15 alatt marad.
A gép a számokat 14 jegyre kerekitve irja ki 0.1-től 99999999999999-ig tortalakban, azonkivül pedig normálalakban.
A szám, illetve a tizes kitevő + előjele elhagyható /kiiráskor a gép is betükoz ir a helyére/, mig a negativ előjelet mindig ki kell tenni.

A VÁLTOZÓK

A duplapontos bővitésben természetesen érvényben maradnak a korábbi változótipusok.
Az új, dupla pontosságú /ezután röviditve D/ változó jele a felkiáltójel /!/. D változó neve egy betüből és a felkiáltójelből áll. Kétbetüs név itt nem használható.

D-tömbök azonositására szintén egy betü használható, ami után a felkiáltójel és a zárójeles indexkifejezés áll. Vektor és mátrix azonositója nem lehet azonot betü, a tömbök indexkifejezésében nem szerepelhet D-változó. Az index maximális értéke 255. Igy összesen 31 D-változó és 31 D-tömb szerepelhet egy programban.

Az egyszerü D-változókat az első hivatkozás által már a gép automatikusan definiálja. Ekkor mind a 26 lehetséges változónak rezervál területet, tehát lefoglal 214 byte-ot a memóriából.
A D-tömböket mindig definiálni kell, nincsen automatikus definiálás a 10 alatti tömbökre sem. A definiálást a DIM utasításnál ismertetjük. Itt említjük még meg, hogy egy n x m felső indexhatárú D-tömb definiálására 6+8 x/n + 1/x /m + 1/ byte-ot használ fel a gép.

Példák:

A! B! ... Z! A!/5/ B!/X,Y/ Q!/5,6/

KIFEJEZÉSEK

Duplapontos /D/ kifejezés az, ami duplapontos utasításokban szerepel /lásd később/.
D-kifejezésekben használhatók a zárójelek, az összes aritmetikai müvelet és reláció.
A logikai müveletek /AND OR NOT/ D-kifejezésekhez nem használhatók.
D-kifejezésekben természetesen szerepelhetnek D és real változók és konstansok is.

A függvények közül a következők állhatnak D-kifejezésekben:

ABS	ATN	COS	EXP
INT	LOG	SIN	SGN
SQR	TAN	VAL	

Nem használhatók az RND, PEEK, FRE függvények! /Ha mégis ezek értékére volna szükség egy D-kifejezésben, akkor előbb egy real változónak kell értéket adni ezekkel a függvényekkel és utána ezt a változót kell beirni a D-kifejezésbe/.

Ha a D-kifejezésben real tipusú érték szerepel, akkor azt a gép D-vé alakitja és azon végzi el a müveleteket és fuggvényeket. Igy számolás közben is mindig D-tipusú részeredmény keletkezik.
A D-kifejezések további tulajdonságai azonosak az alap-BASIC-ben megismert real kifejezésekével.

DUPLAPONTOS UTASITÁSOK

Egy utasitás akkor duplapontos, ha előtte felkiáltójel áll. Ez azt jelenti a gép számára, hogy ebben az utasitásban minden számolást dupla pontossággal kell elvégezni. Az összes real tipusú szám és változó D-tipusúvá konvertálódik, és igy megy végbe a kiértékelés.

Három D-utasitás lehetséges, nevezetesen

=/értékadás/ !DIM !PRINT

ÉRTÉKADÁS

Alakja az alap-BASIC-en túl háromféle lehet:

a/ !A! = D-kifejezés értékadás D változónak.

A D-kifejezés kiértékelése közben minden real tipusú változó vagy Constans D tipusúvá konvertálódik, és ezzel az értékkel folytatódik a számolás.

b/ !A = D-kifejezés értékadás real változónak.

D-kifejezés itt a számolások legvégén real-lé konvertálódik, és ezt az értéket kapja a real változó.

c/ !A! = VAL/A$/ D-értékadás stringből.

Mivel duplapontosságú INPUT nincs, ez a módja D-értékek bevitelének.

```
INPUT "Szog:"; A$ : !A! = VAL/A$/
```

d/ !A$ = STR$/D-kifejezés/ String értékadás.

A$ a D-kifejezés értékének stringképét kapja. Az STR az egyetlen string fv. amelyben előfordulhat D-változó, ezért az egyenlőség után kötelező a STR$, és a D-kifejezés után már csak a bezárójel állhat. /További stringmüveletek nem!/

e/ A PI változó a D-értékét adja.

Mivel az értékadások különböző fajtáival az esetleges Real D, illetve D real konverziók megoldhatók, külön konvertáló utasitások nincsenek.

Real D	D real
!A! = B	!A = B!

DIMENZIONÁLÁS

Alakja !DIM A! /6,2/, B!/5/, C!/Q/...

A DIM előtt felkiáltójel áll, utána pedig a változók listája vesszővel elválasztva. !DIM után csak D-tömböt lehet definiálni, real és D nem keverhető.

A D-tombök indexkifejezésében real változók használhatók /D változó nem!/.

Minden D-tombot definiálni kell, a 10 alatti maximális indexüeket is!

ÖSSZEFOGLALVA:

A duplapontos változók:

1 betusek plusz felkiáltójel
PI a D-értéke

Duplapontos müveletek:

\+ - * /

Duplapontos függvények:

ABS ATN COS EXP INT LOG SIN SGN
SIN TAN VAL

Duplapontos utasitások:

= /értékadás/ !PRINT !DIM

ezek alött felkiáltójel áll, ha utánuk bárhol az utasitásban van legalább egy felkiáltójel.